

Complexity-Aware Scheduling for an LDPC Encoded C-RAN Uplink

Kyle Whetzel and Matthew C. Valenti
Lane Dept. of Comp. Sci. and Elect. Eng.
West Virginia University
Morgantown, WV 26506–6109
Email: valenti@ieee.org

Abstract—Centralized Radio Access Network (C-RAN) is a new paradigm for wireless networks that centralizes the signal processing in a computing cloud, allowing commodity computational resources to be pooled. While C-RAN improves utilization and efficiency, the computational load occasionally exceeds the available resources, creating a computational outage. This paper provides a mathematical characterization of the computational outage probability for low-density parity check (LDPC) codes, a common class of error-correcting codes. For tractability, a binary erasures channel is assumed. Using the concept of *density evolution*, the computational demand is determined for a given ensemble of codes as a function of the erasure probability. The analysis reveals a trade-off: aggressively signaling at a high rate stresses the computing pool, while conservatively backing-off the rate can avoid computational outages. Motivated by this trade-off, an effective computationally aware scheduling algorithm is developed that balances demands for high throughput and low outage rates.

I. INTRODUCTION

Fifth-generation (5G) networks are expected to be much denser and offer higher operating frequencies than their predecessors [1], [2]. The implication of densification is that there will be many more base stations with a higher variability in traffic. As signal processing techniques used in wireless networks get more sophisticated, the digital baseband processors will continue to dominate the price, footprint, and power requirements of a wireless network.

A recent trend in wireless networking has been to consolidate the baseband processing of multiple access points. An early version of such consolidation was distributed antenna systems [3]. A more recent concept is that of C-RAN, whereby the baseband processors of several base stations are colocated in a central processing farm [4]. The benefits of C-RAN are numerous and stem from the sharing of commodity hardware by base stations with diverse processing needs. Moreover, C-RAN allows for sophisticated collaborative signal processing, for instance by performing multiuser detection across base stations [5], by transmitting distributed space-time codes [6] or by performing network MIMO [7], [8].

When processing is shared by a limited pool of resources, there is a chance that the instantaneous processing demands cannot be met by the given resources. When this happens, an outage occurs. Such an outage is similar to an outage caused by channel impairments such as fading or interference. In all such outages, the data will need to be retransmitted, assuming the presence of a hybrid-ARQ retransmission protocol.

In this paper, we explore the computational requirements of a typical wireless uplink. The focus is on the transmission from the mobile unit to the base station, known as the *uplink*. Transmissions are assumed to be encoded with low-density parity check (LDPC) codes, a common class of error-correcting codes. For tractability, a binary erasures channel (BEC) is assumed. Using the concept of *density evolution*, the computational demand is determined for a given ensemble of codes as a function of the erasure probability. The analysis reveals a trade-off: aggressively signaling at a high rate stresses the computing pool, while conservatively backing-off the rate can avoid computational outages. Motivated by this trade-off, an effective computationally aware scheduling algorithm is developed that balances demands for high throughput and low outage rates. Simulation results show the benefits of using such a computationally aware scheduling algorithm.

The remainder of this paper is organized as follows. Section II gives an overview of the C-RAN concept. Section III reviews the salient points of LDPC codes, providing a mathematical framework describing the complexity of LDPC codes when used over a BEC. Section IV discusses how the rate-optimized LDPC codes were identified, and shows their complexity as a function of the erasure probability. Section V discusses the various schedulers that were considered. Section VI provides the results of a simulation-based analysis of the schedulers. Finally, Section VII concludes the paper and discusses potential future work.

II. OVERVIEW OF C-RAN

The components of a cellular base station can be grouped into two main entities: The part that does analog-domain processing (e.g., power amplification and RF circuitry) and the digital *baseband* processor. In a traditional cellular network, these components are packaged together in a single unit per base station. However, as technology improved, the analog components became lighter and cheaper allowing them to be placed close to the antenna at the top of towers. The digital equipment, the so called baseband unit (BBU), was left at the base of towers in an equipment shack. This separation of RF and BBU has several benefits. First and foremost the antenna feed line, which is a major source of loss, can be made very short. Also the baseband equipment can be placed

in a controlled enclosure with temperature control and plentiful power.

Once analog to digital conversion (ADC) is performed atop a tower, the sampled signal can then be fed through a high speed optical cable to the BBU. This link between the RF component and BBU is known as the *fronthaul* link. Common Public Radio Interface (CPRI) is the protocol used for this link. CPRI transmits a constant bit rate up to 12 Gbps. It allows for a link of up to 40 kilometers with a latency of only 0.1 ms. This distance allows BBUs for multiple cells to be consolidated in a so called *baseband hotel*. This lends to cost effectiveness because less infrastructure is needed for delivering power, temperature control, and network links to the BBUs [4]. These baseband hotels also allow for joint processing and the sharing of computing resources. This sharing of resources creates a statistical multiplexing gain. The gain comes from exploiting temporal and spatial traffic fluctuations which are inherent to cellular networks.

Mobile networks are hard real-time systems with tight timing and protocol constraints [9]. In a C-RAN, sampled data must be quickly transmitted over the fronthaul and processed in real time with a hard deadline. It is possible for the capacity of the fronthaul link to not be sufficient to meet these deadlines. There is also finite computing resources at each baseband hotel. It is possible the computational load demanded by the network is greater than the processing resources available. If either of these situations arise, a *computational outage* is said to occur. These computational outages are just as detrimental to wireless systems as outages caused by fading and interference [10].

Of the tasks handled at BBUs, forward error control (FEC) is amongst one of the most computationally intensive [11]. FEC aims to add redundancy to transmitted data such that it can recover from errors occurred during transmission. By adding this redundancy data rate is compromised. A lot of literature has focused on maximizing the code rate to approach the Shannon limit. However, signaling at higher data rates requires more computational resources for decoding. This trade-off becomes very important in C-RAN systems. It is the objective of a C-RAN *scheduler* to select the coding scheme in real time to best meet the demands of the system.

III. THE COMPLEXITY OF LDPC CODES

LDPC codes use an iterative decoder that is defined on a *Tanner Graph* [12]. As such, the decoding complexity depends on the number of decoding iterations and the layout of the graph. The number of decoder iterations can be predicted using a concept known as *density evolution*. For ease of exposition, we focus on density evolution for the BEC in this paper, though the work could be extended to other channels, including binary symmetric channel (BSC) and additive white Gaussian noise (AWGN) channels.

In the BEC, the input may be a data 0 or a data 1, while the output may be a data 0, data 1, or an erasure. With probability ϵ , called the *erasure probability*, the bit is erased, in which case the channel outputs an erasure, while with probability $1 - \epsilon$

the bit is received correctly. When the erasure probability is ϵ , the Shannon capacity is $1 - \epsilon$, meaning that a code exists that enables reliable communication at a rate very close to $R = 1 - \epsilon$.

The Shannon capacity of the erasure channel can be reached by using LDPC codes. The input to an LDPC encoder is a *message* \mathbf{u} of length k bits. The encoder maps each message u to a *codeword* \mathbf{c} of length n bits, where $n > k$. The ratio of message bits to the total number of bits in a codeword, $R = k/n$, is called the *code rate*. The code is denoted by \mathcal{C} , which is the set of all codewords.

A. Describing LDPC Codes

An LDPC code may be described by its *parity-check matrix*, H , which is a rank $(n - k)$ matrix whose rows span the dual code \mathcal{C}^\perp , which is the set of all length- n binary vectors that are orthogonal to every vector $\mathbf{c} \in \mathcal{C}$. The parity check matrix H is usually is full rank, in which case H has $n - k$ rows. Because the rows of H are from \mathcal{C}^\perp , and hence are orthogonal to every codeword, it follows that $\mathbf{c}H^T = 0$ for all $\mathbf{c} \in \mathcal{C}$. An LDPC code is characterized by a sparse parity-check matrix; i.e., a large matrix containing very few ones. The sparseness results in a low decoder complexity, even as the code length becomes long (as is necessary to approach the Shannon capacity).

An example parity-check matrix is as follows:

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

By expanding $\mathbf{c}H^T = 0$, a set of parity-check equations may be found, one for each row of H . For instance, the parity-check equations for H matrix of (1) are:

$$\begin{aligned} c_0 \oplus c_1 \oplus c_2 &= 0 \\ c_3 \oplus c_4 \oplus c_5 &= 0 \\ c_0 \oplus c_3 \oplus c_6 &= 0 \\ c_1 \oplus c_4 \oplus c_7 &= 0 \\ c_2 \oplus c_5 \oplus c_8 &= 0. \end{aligned}$$

Code bit c_j is said to *participate* in the i^{th} parity-check equation if there is a 1 in the $(i, j)^{\text{th}}$ position of H .

A *Tanner graph* is a bipartite graph that represents an H matrix. In Tanner Graphs, the two independent sets of vertices are called the *check nodes* and the *variable nodes*. The check nodes represent the $n - k$ parity-check equations. The variable nodes represent the n code bits. If an entry in the parity-check matrix $H_{i,j} = 1$ then the i^{th} check node is connected to the j^{th} variable node. Thus, the Tanner graph is a pictorial representation showing which code bits participate in which parity-check equations. Fig. 1 shows the Tanner Graph for the parity check matrix given in (1).

Decoding can be performed using the Tanner Graph. First the variable nodes are loaded with the observed code bits. In Fig. 1 an example received message containing erasures is

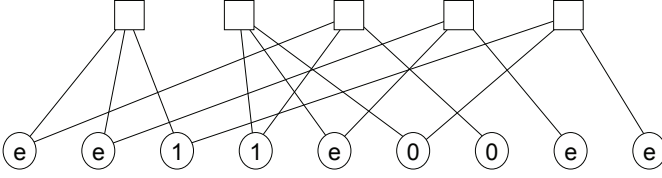


Fig. 1. Example Tanner Graph

shown. Each check node examines its incident edges to see if it is connected to a single variable node containing an erasure. If this is the case those check nodes can then correct that erasure using the logic of a single parity-check code. This process of checking incident edges at each check node is then iterated until all erasures are corrected or there are no more possible corrections to be made.

The *degree* of a check node is the number of incident edges connected to it. This number is equal to the the number of ones, or the *Hamming Weight*, of the corresponding row in the H matrix. The complexity of decoding using a parity-check matrix is dependent on the degree of the check nodes. Therefore, it is desired to have H matrices with low row weight.

If the rows of a LDPC parity-check matrix have constant weight, or constant check-node degree d_c , the code is said to be *check regular*. If the Hamming weight of the columns of a LDPC parity-check matrix are also constant, that is the variable-node degree d_v is constant, the code is said to be *regular*. For brevity, the shorthand (d_v, d_c) is used to specify a regular code with variable-node degree d_c and check-node degree d_c .

Although regular codes are simple and perform moderately well, they are not capable of achieving capacity. Irregular LDPC codes are those without a constant degree, and when properly designed are capable of achieving capacity. When designing irregular LDPC codes, the variable node distribution is not constant. The check node degree is however typically held constant, or close to it.

The design of irregular LDPC codes consists of choosing the proper degree distribution. Let ρ_i denote the fraction of *edges* touching degree i check nodes, and λ_i denote the fraction of *edges* touching degree i variable nodes. The degree distributions can be described in polynomial form. $\rho(x) = \sum_i \rho_i x^{i-1}$ is the distribution for check nodes, and $\lambda(x) = \sum_i \lambda_i x^{i-1}$ for variable nodes. From [12], the rate of the code can be found in terms of the polynomial form of the degree distributions as

$$R = 1 - \frac{\int_0^1 \rho(x) dx}{\int_0^1 \lambda(x) dx}. \quad (2)$$

B. Density Evolution

Density evolution is a tool for predicting the erasure probability as a function of the number of decoder iterations. For a LDPC code over a BEC, [12] states the probability that a variable-node remains erased after the ℓ^{th} iteration is

$$\epsilon_\ell = \epsilon_0 \lambda(1 - \rho(1 - \epsilon_{\ell-1})) \quad (3)$$

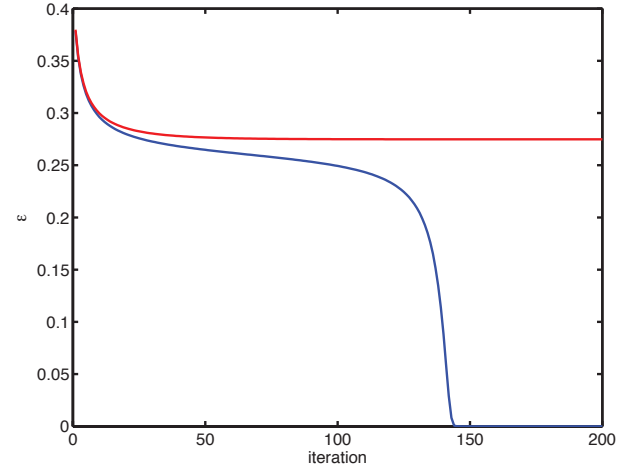


Fig. 2. Density Evolution of a (3,6) regular code. The threshold $\epsilon^* = 0.4294$ for this code. One plot is for $\epsilon_0 = 0.43$ the other $\epsilon_0 = 0.429$

where ϵ_0 is the initial erasure probability before any error correcting has been performed. A code is said to *converge* if the erasure probability is reduced to or below a certain threshold after a sufficiently large number of iterations. Let the *convergence threshold* be denoted by ϵ_{thresh} . The criteria for convergence is that the erasure probability is constantly decreasing; i.e., if $\epsilon_\ell < \epsilon_{\ell-1}$ for all ℓ in which $\epsilon_\ell > \epsilon_{\text{thresh}}$.

From the requirement that $\epsilon_\ell < \epsilon_{\ell-1}$ and (3),

$$\underbrace{\epsilon_0 \lambda(1 - \rho(1 - \epsilon_{\ell-1}))}_{\epsilon_\ell} < \epsilon_{\ell-1}. \quad (4)$$

By performing the change of variable $\epsilon_{\ell-1} \rightarrow x$ and defining the function

$$f(\epsilon_0, x) = \epsilon_0 \lambda(1 - \rho(1 - x)), \quad (5)$$

it is found that decoder convergence is possible if and only if $f(\epsilon_0, x) < x$ for all $\epsilon_{\text{thresh}} \leq x \leq \epsilon_0$. It is useful to find the largest ϵ_0 for which the code converges. This initial erasure probability threshold, will be denoted as ϵ^* . Fig. 2 shows density evolution for a code when the ϵ_0 is just above and just below ϵ^*

Rearranging (5) and using $f(\epsilon_0, x) < x$ yields

$$\epsilon(x) = \frac{x}{\lambda(1 - \rho(1 - x))}. \quad (6)$$

To find ϵ^* the largest value of ϵ_0 such that $\epsilon_0 < \epsilon(x)$ for all x must be determined. The function $\epsilon(x)$ is a convex function, ϵ^* is its minimum value, as follows:

$$\epsilon^* = \min\{\epsilon(x) : \epsilon(x) > x\}. \quad (7)$$

C. LDPC Code Complexity

The complexity of a decoder is the amount of processing resources required to successfully decode a received message. Let K be complexity required per codeword, which is equal to the number of decoder iterations multiplied by the number of edges in a Tanner graph. Assuming a check-regular LDPC code $K = \ell d_c(n - k)$.

To get a better understanding of complexity versus throughput of information it would be beneficial to know the required work per data bit. Let C be complexity per data bit. To find C , K will be normalized by dividing by the number of data bits, k ; i.e., $C = \frac{\ell d_c(n-k)}{k}$. Knowing that $R = k/n$ the complexity per code bit can be written in terms of the code rate as

$$C = \frac{\ell d_c(1-R)}{R}. \quad (8)$$

IV. RATE OPTIMIZED CODES

A palette of codes is required for use by the complexity-aware scheduler. The codes were designed by constraining them to be check-regular with $d_c = 7$ and then finding the variable-node degree $\lambda(x)$ that maximizes ϵ^* . For the code to be valid it must meet the following constraints:

$$0 \leq \lambda_i \leq 1 \quad \text{for } 1 \leq i \leq d_{\max} \quad (9)$$

$$\sum_{i=1}^{d_{\max}} \lambda_i = 1 \quad (10)$$

$$\sum_{i=1}^{d_{\max}} \frac{\lambda_i}{i} = \frac{1}{(d_c)(1-R)} \quad (11)$$

where d_{\max} is the largest allowable variable node degree. It was found that performance is enhanced by avoiding degree-1 variable nodes, which is accomplished by changing (9) to $\lambda_1 = 0$ and $0 \leq \lambda_i \leq 1$ for $2 \leq i \leq d_{\max}$.

With $d_{\max} = 200$, eight codes were developed with rates. Let (R, ϵ^*) denote the rate / erasure threshold pair for each code. The developed codes had (R, ϵ^*) pairs (1/5, 0.728), (1/4, 0.708), (1/3, 0.657), (2/5, 0.589), (1/2, 0.478), (3/5, 0.367), (2/3, 0.274), and (3/4, 0.167). Thus, the largest erasure probability that can be handled by the codes is 0.728. Any situation where the initial erasure probability is greater will result in failure to decode the message.

By using (8), complexity is calculated for various values of ϵ_0 using density evolution described in section III-B. The convergence threshold was set to $\epsilon_{\text{thresh}} = 10^{-3}$ and the maximum number of iterations was set to 10^3 . Initial erasure probability was varied from 0 to just above the threshold ϵ^* for each code. The resulting complexities are shown in Fig. 3. As the initial erasure probability ϵ_0 approaches the threshold ϵ^* for each code, the computational complexity spikes up. This shows the desirability of complexity aware scheduling systems. If operating near ϵ^* the computation load can be lowered dramatically by switching to a lower coding rate.

V. COMPLEXITY AWARE SCHEDULING FOR C-RAN

Four different scheduling algorithms are considered for the C-RAN system. These include complexity unaware and complexity aware methods. Let C_{server} be the total computational complexity resources available to a computing cluster of size N . Let C_{κ} be the computational complexity required at a base station for user κ within the computing cluster. The rate, $r_{\kappa} \in \mathcal{R}$, is the code rate assigned to the user κ . Throughput,

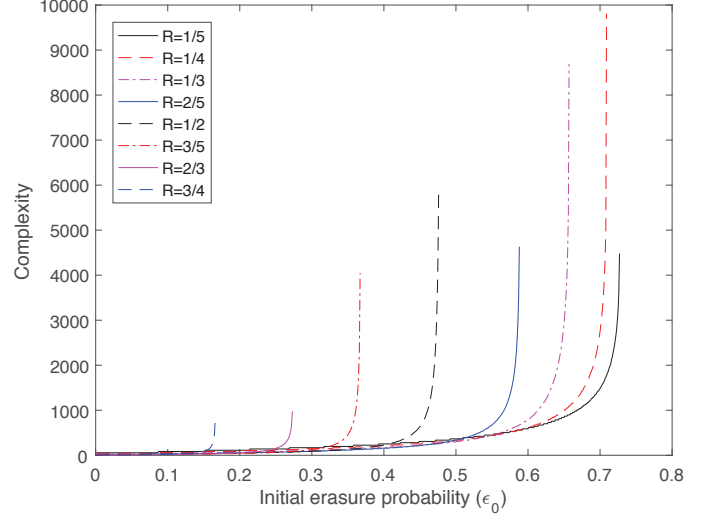


Fig. 3. Complexity Required For Convergence. Note the spike of complexity values near each codes ϵ^* .

T , will be defined as the average of all assigned code rates r_{κ} contained in the computing cluster.

In the **Max Rate Selection (MRS)** algorithm, each user in the cluster κ is assigned the maximum code rate available based on ϵ_0 at the base station and ϵ^* of the codes in the pool. If decoding can't be performed for a certain user, r_{κ} is set to 0. Complexity, C_{κ} , is then calculated for each user within the computing cluster as a batch process. If $\sum_{\kappa=1}^N C_{\kappa} > C_{\text{server}}$ a computational outage occurs. The rate is set to zero for all users resulting in zero throughput.

The **Easiest Job First (EJF)** algorithm initializes by performing the MRS code selection scheme. If $\sum_{\kappa=1}^N C_{\kappa} > C_{\text{server}}$ the scheduler selects the user κ that has the lowest complexity value, or the easiest job. If this minimum complexity $C_{ej} < C_{\text{server}}$ decoding is performed for that user and C_{ej} is then added to the sum of complexities already decoded, C_{sum} . C_{sum} is set to 0 before any decoding has occurred. This process then repeats by selecting the easiest job of those remaining to be decoded as long as $C_{\text{sum}} < C_{\text{server}}$.

The **Local Limit** algorithm starts by selecting the highest possible code rate for each user as in the MRS algorithm. It sets a limit, C_{loc} , on the computational resources allocated to one user. If any user κ demands a decoder complexity greater than C_{loc} , r_{κ} is bumped down to a lower rate code if one exists. Complexity is then recalculated. This process is then reiterated until $\sum_{\kappa=1}^N C_{\kappa} < C_{\text{server}}$, or there are no lower rate codes to recede to. If the case is the latter, and no more codes are available, a computational outage has occurred.

The **Scheduling with Complexity Cutoff (SCC)** algorithm also initializes by setting rates to the maximum possible as in the MRS scheme. It then checks if $\sum_{\kappa=1}^N C_{\kappa} > C_{\text{server}}$. If that is the case user κ^* is chosen such that C_{κ^*} is the maximum C_{κ} . The code rate for user κ^* , r_{κ^*} , is then decreased to the next lower coding rate if one exists. If a lower rate code does not exist for κ^* the user κ with the next highest C is then

chosen. $\sum_{\kappa=1}^N C_{\kappa}$ is then recalculated with the new coding rate. If this sum is less than C_{server} decoding commences, else the process is recursed such that an updated κ^* is selected. The recursion proceeds until the complexity constraint is met or no lower rate codes are available.

VI. SIMULATION RESULTS

To illustrate the efficacy of the scheduling algorithms, a simulation study was conducted. In the simulations, 110 base stations are placed in a hexagonal grid pattern. Mobile units (MU) are randomly dropped into the system such that there is at most only one active mobile per base station cell. The number of cells with users is the channel *utilization*. The computing cluster will be a central group of adjacent cells which have pooled computing resources.

Let $\{X_i\}$ indicate the set of placed MU's and their locations. A given MU X_i is connected to a particular base station, denoted Y_i . After each set of MU's are placed, the signal-to-interference-and-noise ratio (SINR) is calculated for each base station in the computing cluster. The SINR at basestation Y_j is

$$\gamma_j = \frac{|Y_j - X_j|^{\alpha(s-1)}}{\Gamma^{-1} + \sum_{i \neq j} |Y_j - X_i|^{-\alpha} |Y_i - X_i|^{s\alpha}} \quad (12)$$

where by $|Y_j - X_i|$ being the distance from a base station Y_j MU X_i , α is the path-loss exponent, s is the compensation factor for fractional power control, Γ is the SNR at unit distance, and the summation is over all interferers. The power control factor is set to $s = 0.1$, which is the value that maximizes throughput [13].

Once the SINR is calculated for each base station in the computing cluster, the corresponding erasure probability is found by using the complementary cumulative distribution function (CCDF) of the SINRs obtained. The CCDF makes use of the SINRs obtained with all of the default simulation values. By definition, the CDF of a random variable X evaluated at x is the probability that X is less than x . That is $F_X(x) = P(X < x)$. Since this is a probability, as is ϵ_0 , the value ranges from 0 to 1. Because high SINR values need to be mapped to low initial erasure probability ϵ_0 values, the CCDF is used. The complementary cumulative distribution function $\text{CCDF} = 1 - \text{CDF}$. An exponential curve is then fitted to the CCDF.

Unless otherwise specified, the simulations used the default values of the simulation parameters are as follows: path-loss exponent $\alpha = 3$, utilization $u = 100\%$, cluster size $N = 7$, SNR at unit distance $\Gamma = 20$ dB. The number of Monte Carlo trials is set to 10^5 .

The first simulation explores the impact of path-loss exponent α as the independent variable. It was varied such that $2 \leq \alpha \leq 5$, with $\alpha = 2$ corresponding to free-space and larger path-losses corresponding to rough terrain or blockage. Fig. 4 shows the effect of varying the pathloss exponent α on throughput for the different scheduling algorithms. When $\alpha = 2$, the throughput for all schedulers is very low due to the

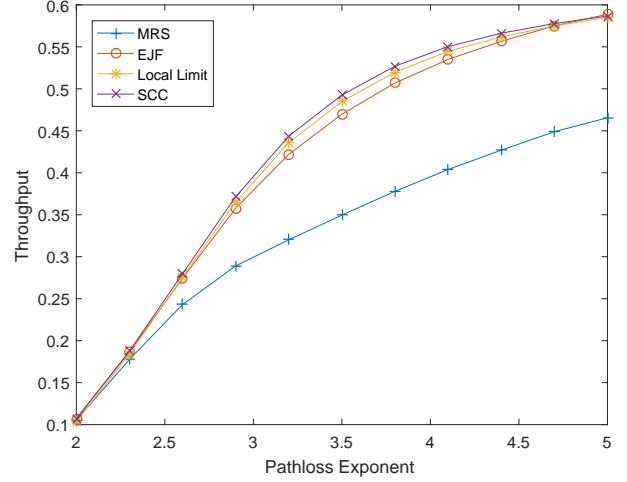


Fig. 4. Throughput versus Path-loss exponent α . α is varied from a value for free space, 2, to 5 which corresponds to very rough terrain or high blockage. The computing cluster size is held constant at 7 and the channel utilization constant at 100%.

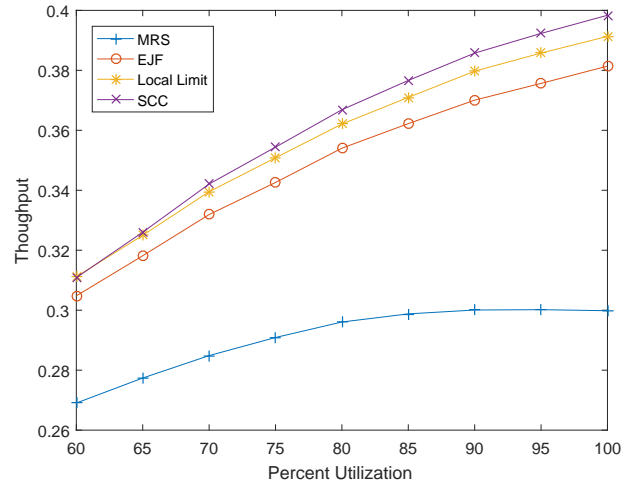


Fig. 5. Throughput versus Channel Utilization. If there is no user in a computing cluster cell the throughput for that cell is set to zero. Pathloss exponent α is held constant at 3 and the computing cluster size at 7.

high interference, which does not get sufficiently attenuated in free space.

The second simulation explored the effect of channel utilization. The channel usage was varied between $0.6 \leq u \leq 1$. Fig. 5 displays the throughput when varying the channel utilization. When randomly placing the mobile units there is no guarantee a user will be placed in every cell of the computing cluster. The throughput is lower at lower channel utilization's because no data is being transmitted if there isn't a user in that cell.

The last simulation explored the effect of the computing cluster size, by varying in the range $1 \leq N \leq 10$. Fig. 6 shows the throughput when varying the computing cluster size. The first data point here is when cluster size is set to one. This is the same as a traditional cellular network where no processing

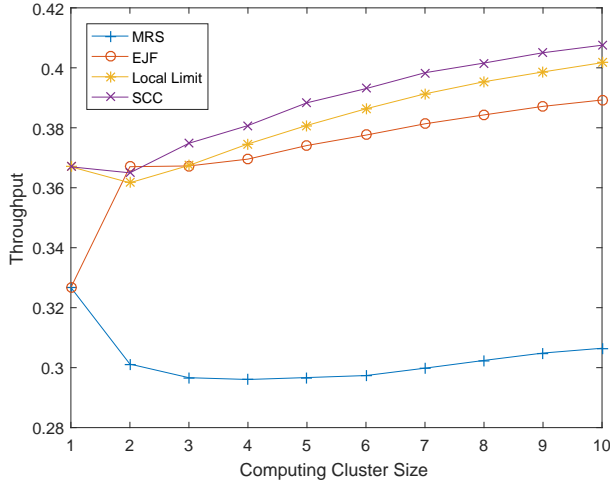


Fig. 6. Throughput versus the Computing Cluster Size. The pathloss exponent is constant at $\alpha = 3$ and the channel utilization is held constant at 100%.

resources are shared. From the plot you can see that increasing cluster size increases the throughput for all complexity aware algorithms.

Looking at all the simulations it can be seen that the complexity aware schemes always outperform the complexity unaware MRS. In terms of throughput, the more sophisticated SCC does as good or better for all values of the variables being varied.

VII. CONCLUSIONS

Advancements in technology has made it feasible to process baseband signals of wireless networks in a central processing pool. This network paradigm shift, known as C-RAN, creates many benefits. These include the ability to share computing resources among multiple cells and centralizing the management of system infrastructure. These benefits come along with some challenges. The primary being the load imposed on the front haul connection, and the risk that the demand on computing resources exceeds its capacity. When either the fronthaul capacity isn't sufficient, or the processing center doesn't have sufficient computation resources a outage occurs. From a provider's perspective this outage is no different from a more traditional impairment such as fading or interference.

This paper has provided an analysis of the computational requirements of a C-RAN uplink, and has developed and analyzed computationally aware scheduling algorithms for

C-RAN. In order to perform a mathematically precise and tractable analysis, we have assumed that the transmissions are encoded with LDPC codes and sent over a BEC channel. While this allows for the per-bit complexity to be accurately predicted, future work should extend the analysis to other kinds of channels.

Another possible future direction is to create LDPC codes that are themselves complexity aware; i.e., designed with an additional constraint that attempts to reduce the complexity rather than just minimize the convergence threshold. Finally, other, more sophisticated complexity aware algorithms could be developed.

REFERENCES

- [1] F. Boccardi, R. W. Heath Jr, A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *Communications Magazine, IEEE*, vol. 52, no. 2, pp. 74–80, 2014.
- [2] D. Torrieri, S. Talarico, and M. C. Valenti, "Analysis of a frequency-hopping millimeter-wave cellular uplink," *IEEE Transactions on Wireless Communications*, vol. 15, pp. 7089–7098, Oct 2016.
- [3] A. A. Saleh, A. Rustako, and R. Roman, "Distributed antennas for indoor radio communications," *IEEE Transactions on Communications*, vol. 35, no. 12, pp. 1245–1251, 1987.
- [4] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for Mobile Networks-A Technology Overview," *IEEE Communications Surveys Tutorials*, vol. 17, pp. 405–426, Firstquarter 2015.
- [5] M. C. Valenti and B. D. Woerner, "Iterative multiuser detection, macro-diversity combining, and decoding for the TDMA cellular uplink," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 1570–1583, Aug 2001.
- [6] Y. Tang and M. C. Valenti, "Coded transmit macrodiversity: Block space-time codes over distributed antennas," in *Proceedings IEEE VTS 53rd Vehicular Technology Conference*, vol. 2, pp. 1435–1438, 2001.
- [7] S. Park, C. B. Chae, and S. Bahk, "Large-scale antenna operation in heterogeneous cloud radio access networks: a partial centralization approach," *IEEE Wireless Communications*, vol. 22, pp. 32–40, June 2015.
- [8] Y. Zhou and W. Yu, "Fronthaul compression and transmit beamforming optimization for multi-antenna uplink C-RAN," *IEEE Transactions on Signal Processing*, vol. 64, pp. 4138–4151, Aug 2016.
- [9] M. C. Valenti, S. Talarico, and P. Rost, "The Role of Computational Outage in Dense Cloud-Based Centralized Radio Access Networks," in *IEEE Global Conference on Communications*, (Austin (TX), USA), December 2014.
- [10] P. Rost, S. Talarico, and M. C. Valenti, "The Complexity Rate Tradeoff of Centralized Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 14, pp. 6164–6176, Nov 2015.
- [11] P. Rost, A. Maeder, M. C. Valenti, and S. Talarico, "Computationally Aware Sum-Rate Optimal Scheduling for Centralized Radio Access Networks," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2015.
- [12] C. Schlegel, *Trellis and Turbo Coding*. Piscataway, NJ Hoboken, NJ: IEEE Press Wiley-Interscience, 2004.
- [13] M. Coupechoux and J. M. Kelif, "How to set the Fractional Power Control Compensation Factor in LTE," in *34th IEEE Sarnoff Symposium*, May 2011.